

# A combinatorial algorithm for constrained assortment optimization under nested logit model

Tian Xie<sup>1</sup>

*Research Center for Management Science and Information Analytics, School of Information Management and Engineering,  
Shanghai University of Finance and Economics, China*

---

## Abstract

We consider the assortment optimization problem with disjoint-cardinality constraints under two-level nested logit model. To solve this problem, we first identify a candidate set with  $O(mn^2)$  assortments and show that at least one optimal assortment is included in this set. Based on this observation, a fast algorithm, which runs in  $O(mn^2 \log mn)$  time, is proposed to find an optimal assortment.

*Keywords:* Revenue management, Assortment optimization, Nested logit model, Combinatorial algorithm

---

## 1. Introduction

Assortment optimization is an important topic in revenue management. Briefly speaking, it considers the situation that a decision maker wants to determine a set of offered products so as to maximize the expected revenue. In this case, each product is assigned with an exogenous fixed price and a consumer chooses at most one item from the available products according to some preference, which is specified by certain choice model. Of course, the optimal set is usually not simply the entire product set, and the optimal structure is highly dependent on the underlying choice model. The interested readers are referred to the excellent survey [3] for the related literature and applications.

In practice, we often encounter the case that the number of offered products cannot exceed certain threshold although it is indeed beneficial to offer a larger assortment, due to the limited resource. For example, shelf space in a shop may be limited, or the number of advertisement displayed on a webpage is subject to the size of the screen. Thus, it is reasonable to consider the problem with a constraint on the total size of the offer set. If all the products are assumed to be the same size, that is only the number of products matters. Then, the constraints imposed are also referred to as cardinality constraints.

The most popular choice model is the so-called multinomial logit (MNL) model (see [7]). Talluri and van Ryzin [10] first introduced this model to revenue management. They considered the associated unconstrained assortment optimization and showed that the optimal solution is among the assortments with

---

*Email address:* xietiansh@gmail.com (Tian Xie)

*revenue-ordered* structure. [9] considered the MNL problem with cardinality constraints and proposed an strongly polynomial algorithm to generate an  $O(nC)$ -sized candidate assortment set which contains at least one optimal assortment, where there are  $n$  products and the shop can offer no more than  $C$  products. Although MNL model demonstrates clear advantages by providing tractability in many assortment optimization problems, it is criticized a lot due to the independence of irrelevant alternatives (IIA) property (see [1] for more information).

The two-level nested logit model (see [11, 8]) is an extension of MNL model and partially alleviates the drawbacks of IIA. This model divides products into several nests and describes a consumer's behavior as firstly choosing a nest and then picking a product in the chosen nest. [2] showed by imposing some additional conditions, the assortment optimization under two-level nested logit model is polynomial time solvable. Note that the products in the same nest must have some similarity. If all the products in the same nest are assumed to be the same size, the decision maker can plan how many products to offer for each nest before determining the optimal assortment. The constraints are referred as *disjoint-cardinality constraints*. Furthermore, if we assume all the products are of the same size, the decision maker only need to plan how many products to offer in total. The constraints are referred as *joint-cardinality constraints*. The linear program formulation for the unconstrained, disjoint-cardinality constrained, and joint-cardinality constrained assortment optimization problems were provided in [2, 5, 4] respectively. By exploring more properties of the unconstrained problem, a greedy strongly polynomial time algorithm was given in [6].

In this paper, we consider the assortment optimization problem with disjoint-cardinality constraints under two-level nested logit model. The previous work [5] formulated the problem into a linear program with  $(m + 1)$  decision variables and  $O(mn^2)$  constraints, However, it is well known that whether the linear programming is strongly polynomial time solvable is still open. The major contribution of this paper is that we construct a candidate assortments set which has  $O(mn^2)$  elements and contains at least one optimal assortment, where  $m$  is the number of nests and  $n$  is the number of products in each nest. Based on this, we design a combinatorial algorithm to find an optimal assortment which runs in  $O(mn^2 \log mn)$  time. To the best of our knowledge, our algorithm demonstrates the least computational complexity comparing to other known algorithms. Our computational experiments also suggest that our approach is much more efficient than directly solving the linear program in [5] by CPLEX.

We describe the problem and the linear programming formulation in Section 2. We derive optimality conditions and identify a candidate set with  $O(mn^2)$  assortments containing at least one optimal assortment in Section 3, followed by the algorithm to find an optimal assortment in Section 4. We present some computational results in Section 5 to prove the efficiency of our method.

## 2. Review of the problem

### 2.1. Problem description

We start off by briefly describing the mathematical formulation in [2] of the two-level nested logit model.

Suppose there are  $m$  nests indexed by  $\{1, 2, \dots, m\}$ . In each nest, there are  $n$  products and denote the  $j$ -th product in the  $i$ -th nest as  $ij$ . (Note that there are no overlapping between products in different nests.) A consumer firstly chooses some nest  $i$  with probability  $\mathbb{P}(\mathcal{I} = i)$ , and then picks product  $ij$  with probability  $\mathbb{P}(\mathcal{J} = j | \mathcal{I} = i)$ . There is also a no-purchase option indexed by 0 which represents the case that the consumer leaves without making any purchase and the probability is denoted as  $\mathbb{P}(\mathcal{I} = 0)$ . It is assumed that the no-purchase option is isolated from the nests, i.e., if a consumer has chosen nest  $i$ , she will definitely buy a product. We assign each product  $ij$  with weight  $v_{ij} \geq 0$  and revenue  $r_{ij} \geq 0$ , while the no-purchase option has weight  $v_0 > 0$  and revenue 0. In addition, each nest  $i$  has a dissimilarity parameter  $\gamma_i \in (0, 1]$ . Without loss of generality, we assume every nest includes exact  $n$  products, otherwise we simply add dummy products with weight 0.

An assortment is a bundle of sets  $S = (S_1, S_2, \dots, S_m)$  where  $S_i \subseteq \{i1, i2, \dots, in\}$  is the set of products that offered in nest  $i$ . Therefore, the final offer set is  $\cup_{i=1}^m S_i$ . Given the assortment  $(S_1, S_2, \dots, S_m)$ , the customer chooses nest  $i$  with probability

$$\mathbb{P}(\mathcal{I} = i) = \frac{V_i(S_i)^{\gamma_i}}{v_0 + \sum_{k=1}^m V_k(S_k)^{\gamma_k}}, \quad \mathbb{P}(\mathcal{I} = 0) = \frac{v_0}{v_0 + \sum_{k=1}^m V_k(S_k)^{\gamma_k}},$$

where  $V_i(S_i) = \sum_{j \in S_i} v_{ij}$  is the total weight of products offered in nest  $i$ ; the customer picks product  $ij$  conditioning on choosing nest  $i$  with probability

$$\mathbb{P}(\mathcal{J} = j | \mathcal{I} = i) = \frac{v_{ij}}{V_i(S_i)}.$$

Therefore, the expected revenue is given by

$$\Pi(S_1, \dots, S_m) = \sum_{i=1}^m \sum_{j=1}^n r_{ij} \mathbb{P}(\mathcal{J} = j | \mathcal{I} = i) \mathbb{P}(\mathcal{I} = i) = \frac{\sum_{i=1}^m V_i(S_i)^{\gamma_i} R_i(S_i)}{v_0 + \sum_{i=1}^m V_i(S_i)^{\gamma_i}}$$

where  $R_i(S_i) = \frac{\sum_{j \in S_i} r_{ij} v_{ij}}{V_i(S_i)}$  is the weighted average revenue of products offered in nest  $i$ .

The goal of assortment optimization is to maximize the expected revenue. In this paper, we consider the case that  $S$  has disjoint-cardinality constraints. That is there is a size limitation  $C_i$  for the offer set provided in nest  $i$ . Mathematically, the corresponding assortment optimization can be formulated as

$$\max_{(S_1, S_2, \dots, S_m): \forall i, |S_i| \leq C_i} \frac{\sum_{i=1}^m V_i(S_i)^{\gamma_i} R_i(S_i)}{v_0 + \sum_{i=1}^m V_i(S_i)^{\gamma_i}}. \quad (1)$$

## 2.2. The linear programming formulation

[5] showed that the problem (1) can be written as a linear program. It is clear that for every assortment  $S = (S_1, S_2, \dots, S_m)$  and the corresponding expected revenue  $Z(S)$ , we have

$$Z(S) = \Pi(S_1, \dots, S_m) = \frac{\sum_{i=1}^m V_i(S_i)^{\gamma_i} R_i(S_i)}{v_0 + \sum_{i=1}^m V_i(S_i)^{\gamma_i}} \Leftrightarrow v_0 Z = \sum_{i=1}^m V_i(S_i)^{\gamma_i} (R_i(S_i) - Z). \quad (2)$$

Therefore, for any  $z > \Pi(S_1, \dots, S_m)$ , we have  $v_0 z > \sum_{i=1}^m V_i(S_i)^{\gamma_i} (R_i(S_i) - z)$ ; for any  $z < \Pi(S_1, \dots, S_m)$ ,  $v_0 z < \sum_{i=1}^m V_i(S_i)^{\gamma_i} (R_i(S_i) - z)$ . Consequently, given  $(S_1, S_2, \dots, S_m)$ , one has  $Z(S) = \min\{z : v_0 z \geq \sum_{i=1}^m V_i(S_i)^{\gamma_i} (R_i(S_i) - z)\}$ .

Denote  $(S_1^*, S_2^*, \dots, S_m^*)$  as the optimal assortment and  $Z^*$  as the associated expected revenue. Based on above discussion,

$$\begin{aligned} Z^* &= \max_{(S_1, S_2, \dots, S_m) : \forall i, |S_i| \leq C_i} \min \left\{ z : v_0 z \geq \sum_{i=1}^m V_i(S_i)^{\gamma_i} (R_i(S_i) - z) \right\} \\ &= \min \left\{ z : v_0 z \geq \sum_{i=1}^m V_i(S_i)^{\gamma_i} (R_i(S_i) - z) \quad \forall |S_i| \leq C_i, \forall i \right\} \\ &= \min \left\{ z : v_0 z \geq \sum_{i=1}^m \max_{S_i : |S_i| \leq C_i} V_i(S_i)^{\gamma_i} (R_i(S_i) - z) \right\} \end{aligned} \quad (3a)$$

$$= \min \left\{ z : v_0 z \geq \sum_{i=1}^m y_i, y_i \geq V_i(S_i)^{\gamma_i} (R_i(S_i) - z) \quad \forall |S_i| \leq C_i, \forall i \right\} \quad (3b)$$

Therefore, we arrive at a linear program formulation (3b) with  $(m+1)$  decision variables and exponential numbers of constraints. [5] have reduced the size of constraints to polynomial, which will be discussed in Section 3.1.

## 3. The Optimality Conditions

Section 3.1 firstly describe the idea of [5] that reduces exponential constraints in (3b) to  $O(mn^2)$  constraints, which is referred as *optimality conditions for individual nests* in this paper. Based on this idea, we extend the condition to the entire system by combining all optimality conditions for individual nests. We construct a set of  $O(mn^2)$  assortments and claim that this set contains at least one optimal assortment. Moreover, we establish the connection between this set and a piecewise-linear function, where the root of piecewise-linear function is the maximal expected revenue.

### 3.1. The optimality condition for an individual nest

Recall that  $(S_1^*, \dots, S_n^*)$  as the optimal assortment and  $Z^*$  is the maximum expected revenue. Denote  $u_i^* = \max\{Z^*, \gamma_i Z^* + (1 - \gamma_i) R_i(S_i^*)\}$ . [5] showed that when  $\gamma_i \in (0, 1]$ ,

$$V_i(\hat{S}_i)(R_i(\hat{S}_i) - u_i^*) \geq V_i(S_i)(R_i(S_i) - u_i^*), \forall |S_i| \leq C_i \Rightarrow V_i(\hat{S}_i)^{\gamma_i} (R_i(\hat{S}_i) - Z^*) \geq V_i(S_i)^{\gamma_i} (R_i(S_i) - Z^*), \forall |S_i| \leq C_i.$$

For nest  $i$ , according to (3a), it surries to focus on  $\max_{S_i: |S_i| \leq C_i} V_i(S_i)(R_i(S_i) - u_i^*)$ . Although the exact values of  $u_i^*$  are unknown, we can still solve the following problem

$$\max_{S_i: |S_i| \leq C_i} V_i(S_i)(R_i(S_i) - u) \quad (4)$$

with all  $u \in \mathbf{R}$  and find the corresponding optimal solution  $\hat{S}_i(u)$  (if multiple solutions exist, just pick one for each  $u$  is sufficient). The advantage of doing this is that if we set

$$\mathcal{T}_i := \{\hat{S}_i(u), -\infty < u < \infty\}, \quad (5)$$

then  $\hat{S}_i(u^*) \in \mathcal{T}_i$ . Therefore

$$\max_{S_i: |S_i| \leq C_i} V_i(S_i)^{\gamma_i}(R_i(S_i) - Z^*) = \max_{S_i \in \mathcal{T}_i} V_i(S_i)^{\gamma_i}(R_i(S_i) - Z^*). \quad (6)$$

We can find at least one optimal assortment  $S = (S_1, \dots, S_m)$  such that  $S_i \in \mathcal{T}_i$ ,  $\forall i$  and achieve the maximal expected revenue. Thus, we name  $\mathcal{T}_i$  as the *candidate set* for nest  $i$ .

Denote  $x_{ij}$  as the dummy indicator of whether  $ij \in S_i$ . Given  $u$ , the problem (4) for nest  $i$  can be rephrased as

$$\max \left\{ \sum_{j=1}^n v_{ij}(r_{ij} - u)x_{ij} : \sum_{j=1}^n x_{ij} \leq C_i, x_{ij} \in \{0, 1\} \right\}$$

Then the optimal solution of the above problem can be specified as:  $x_{ij}(u) = 1$  iff  $v_{ij}(r_{ij} - u) \geq 0$  and  $v_{ij}(r_{ij} - u)$  is in the largest  $C_i$  elements among  $\{v_{ij}(r_{ij} - u)\}_{j=1}^n$ ;  $x_{ij}(u) = 0$  otherwise. Note that if ties exist, we can rank them arbitrarily. Then a candidate for nest  $i$  can be constructed as:  $\hat{S}_i(u) = \{ij : x_{ij}(u) = 1\}$ .

Next we discuss the size of  $\mathcal{T}_i$ . Consider the following  $n + 1$  lines:  $f_0(u) = 0$ ,  $f_j(u) = v_{ij}(r_{ij} - u)$ . To construct  $\hat{S}_i(u)$ , we can simply choose  $C_i$  lines with highest  $f_j(u)$ , and drop all lines below  $f_0(u)$ . There are  $q \leq \frac{1}{2}n(n + 1)$  crosspoints for those  $(n + 1)$  lines, and the corresponding horizontal coordinate is denoted as  $I_1 \leq I_2 \leq \dots \leq I_q$ . For any consecutive  $I_k < I_{k+1}$  with different horizontal coordinate, the relative order of all lines don't change between  $u \in (I_k, I_{k+1})$  because for any  $j', j''$ , the relative order  $f_{j'}(u)$  and  $f_{j''}(u)$  don't change between  $u \in (I_k, I_{k+1})$ . There could be ties on some  $u = I_k$ , and either  $u = I_k - \varepsilon$  or  $u = I_k + \varepsilon$  ( $\varepsilon \rightarrow 0$ ) can be the alternative tie-breaking rule. Therefore, a set  $\mathcal{T}_i$  with no more than  $O(n^2)$  candidates is sufficient for any nest  $i$ . Note that now (3b) can reduced to

$$Z^* = \min \left\{ z \left| v_0 z \geq \sum_{i=1}^m y_i, y_i \geq V_i(S_i)^{\gamma_i}(R_i(S_i) - z) \forall S_i \in \mathcal{T}_i, \forall i \right. \right\} \quad (7)$$

which is a linear program with  $(m + 1)$  decision variables and  $O(mn^2)$  constraints. We refer interested readers to [5] for more details.

### 3.2. The optimality condition of the whole problem

To determine the exact optimal solution, the characterization of all the optimal assortments is stated in Proposition 1.

**Proposition 1** (Optimal Assortments). *Denote  $Z^*$  as the maximal expected revenue. An assortment  $S = (S_1, \dots, S_m)$  is an optimal assortment if and only if*

$$S_i \in \arg \max_{S_i: |S_i| \leq C_i} \{V_i(S_i)^{\gamma_i} (R_i(S_i) - Z^*)\}, \forall i.$$

*Proof.* Since  $Z^*$  is the maximal expected revenue, by the definition of expected revenue, it must be the case that

$$Z(S) - Z^* = \frac{\sum_{i=1}^m V_i(S_i)^{\gamma_i} (R_i(S_i) - Z^*) - v_0 Z^*}{v_0 + \sum_{i=1}^m V_i(S_i)^{\gamma_i}} \leq 0,$$

which is equivalent to  $\sum_{i=1}^m V_i(S_i)^{\gamma_i} (R_i(S_i) - Z^*) - v_0 Z^* \leq 0$ . For any optimal assortment  $S^* = (S_1^*, \dots, S_m^*)$ ,  $Z(S^*) - Z^* = 0$ , which is equivalent to  $\sum_{i=1}^m V_i(S_i^*)^{\gamma_i} (R_i(S_i^*) - Z^*) - v_0 Z^* = 0$ . If  $S_i$  maximizes  $V_i(S_i)^{\gamma_i} (R_i(S_i) - Z^*)$  for all  $i$ ,

$$0 \geq \sum_{i=1}^m V_i(S_i)^{\gamma_i} (R_i(S_i) - Z^*) - v_0 Z^* \geq \sum_{i=1}^m V_i(S_i^*)^{\gamma_i} (R_i(S_i^*) - Z^*) - v_0 Z^* = 0,$$

which implies that  $S$  is also an optimal assortment. Otherwise, if there exists some  $j$  that  $V_j(S_j)^{\gamma_j} (R_j(S_j) - Z^*) < V_j(S_j')^{\gamma_j} (R_j(S_j') - Z^*)$ , then

$$\sum_{i=1}^m V_i(S_i)^{\gamma_i} (R_i(S_i) - Z^*) - v_0 Z^* < V_i(S_i')^{\gamma_i} (R_i(S_i') - Z^*) + \sum_{i \neq j} V_i(S_i)^{\gamma_i} (R_i(S_i) - Z^*) - v_0 Z^* \leq 0,$$

which implies that  $S$  is not optimal.  $\square$

Note that optimal assortment may not be unique. In some extreme cases, the optimal set can include exponential number of elements. However, to find one optimal assortment is easy. In particular, (6) and Proposition 1 show that the assortment  $(S_1, \dots, S_m)$  where  $S_i \in \arg \max_{S_i \in \mathcal{T}_i} \{V_i(S_i)^{\gamma_i} (R_i(S_i) - Z^*)\}$  is optimal. The difficulty is that the exact value of  $Z^*$  is still unknown. Next, we want to find  $Z^*$  through the linear program formulation.

**Proposition 2** (Necessary condition). *All optimal solutions  $(Z^*, \{y_1^*, \dots, y_m^*\})$  for (3b) satisfy*

$$\begin{aligned} v_0 Z^* &= \sum_{i=1}^m y_i^* \\ y_i^* &= V_i(S_i^*)^{\gamma_i} (R_i(S_i^*) - Z^*) \geq V_i(S_i)^{\gamma_i} (R_i(S_i) - Z^*), \forall |S_i| \leq C_i, \forall i \end{aligned}$$

for some  $(S_1^*, \dots, S_m^*)$  such that  $|S_i| \leq C_i, \forall i$ .

*Proof.* Prove by contradiction. Suppose there is an optimal solution  $(Z^*, \{y_1^*, \dots, y_m^*\})$  where for some  $i$ ,  $y_i^* = V_i(S_i^*)^{\gamma_i} (R_i(S_i^*) - Z^*) \geq V_i(S_i)^{\gamma_i} (R_i(S_i) - Z^*), \forall S_i \in \mathcal{T}_i$ , but  $v_0 Z^* > \sum_{i=1}^m y_i^*$ . Plug  $y_i^*$  in, we have  $v_0 Z^* > \sum_{i=1}^m V_i(S_i^*)^{\gamma_i} (R_i(S_i^*) - Z^*)$ . Recall that  $v_0 > 0$  and all  $v_{ij}$  are non-negative. There always exists  $\varepsilon > 0$  such that  $v_0(Z^* - \varepsilon) \geq \sum_{i=1}^m V_i(S_i^*)^{\gamma_i} (R_i(S_i^*) - Z^* + \varepsilon)$  and hence it is possible to reduce  $Z^*$  to  $Z^* - \varepsilon$ . The solution is not optimal.

Suppose there is an optimal solution  $(Z^*, \{y_1^*, \dots, y_m^*\})$  where for some  $i$ ,  $y_i^* > V_i(S_i)^{\gamma_i}(R_i(S_i) - Z^*)$ ,  $\forall S_i \in \mathcal{T}_i$ , then reducing the corresponding  $y_i^*$  to  $\max_{S_i \in \mathcal{T}_i} V_i(S_i)^{\gamma_i}(R_i(S_i) - Z^*)$  will also make  $(Z^*, \{y_1^*, \dots, y_{i-1}^*, y_i^*, y_{i+1}^*, y_m^*\})$  feasible and hence optimal. For this case,  $v_0 Z^* > \sum_{i=1}^m y_i^*$ . In the case above, the solution is not optimal.  $\square$

Proposition 2 shows that the optimal objective for (3b) must belong to the set

$$\mathcal{Z}^* = \left\{ z : v_0 z = \sum_{i=1}^m \max_{S_i: |S_i| \leq C_i} V_i(S_i)^{\gamma_i}(R_i(S_i) - z) \right\}.$$

If we construct  $\{\mathcal{T}_i\}_{i=1, \dots, m}$  in the form of (5), by (6) the set is equivalent to

$$\mathcal{Z}^* = \left\{ z : v_0 z = \sum_{i=1}^m \max_{S_i \in \mathcal{T}_i} V_i(S_i)^{\gamma_i}(R_i(S_i) - z) \right\}.$$

For each nest  $i$ , we focus on the following function

$$g_i(z) = \max_{S_i \in \mathcal{T}_i} V_i(S_i)^{\gamma_i}(R_i(S_i) - z).$$

Obviously, the epigraph of  $g_i(\cdot)$  is a intersection of  $|\mathcal{T}_i|$  halfplanes, so  $g_i(\cdot)$  is continuous, decreasing (all  $V_i(S_i) \geq 0$ ), convex, piecewise-linear and has at most  $|\mathcal{T}_i| - 1$  breakpoints. Summing up  $\{g_i(\cdot)\}_{i=1}^m$  and  $-v_0 z$  yields

$$G(z) = -v_0 z + \sum_{i=1}^m \max_{S_i \in \mathcal{T}_i} V_i(S_i)^{\gamma_i}(R_i(S_i) - z).$$

**Lemma 1.**  $G(z)$  is a strictly decreasing piecewise-linear convex function on  $z$  with at most  $\sum_{i=1}^m (|\mathcal{T}_i| - 1) = O(mn^2)$  breakpoints.

*Proof.* Firstly,  $g_0(z) := -v_0 z$  is a strictly decreasing linear function. For  $i = 1, 2, \dots, n$ ,  $g_i(z)$  is a continuous piecewise-linear convex function with at most  $|\mathcal{T}_i| - 1$  breakpoints. Between any two consecutive breakpoints,  $g_i(z) = V_i(S_i)^{\gamma_i} R_i(S_i) - V_i(S_i)^{\gamma_i} z$  is decreasing on  $z$  for all  $i$ . In summary,  $G(z)$  is strictly decreasing on  $\mathbf{R}$ . Therefore,  $G(z) = \sum_{i=0}^m g_i(z)$  is a strictly decreasing piecewise-linear convex function on  $z$  with at most  $\sum_{i=1}^m (|\mathcal{T}_i| - 1)$  breakpoints.  $\square$

**Proposition 3** (Necessary condition is sufficient).  $\mathcal{Z}^*$  is a singleton.

*Proof.*  $\mathcal{Z}^*$  is essentially  $\{z : G(z) = 0\}$ . Since  $G(z)$  is strictly decreasing while  $G(-\infty) = \infty$  and  $G(\infty) = -\infty$ , there must be a unique  $z$  such that  $G(z) = 0$ , which implies that  $\mathcal{Z}^*$  is a singleton.  $\square$

By Proposition 3,  $G(z) = 0$  uniquely defines  $Z^*$ , so  $Z^*$  should be the unique element in  $\mathcal{Z}^*$ . The analysis above shows that  $(S_1, \dots, S_m)$  where  $S_i \in \arg \max_{S_i \in \mathcal{T}_i} \{V_i(S_i)^{\gamma_i}(R_i(S_i) - Z^*)\}$  is an optimal assortment.

Before ending this section, we show that our optimality condition can be explained as the existence of a collection of  $O(mn^2)$  candidate assortments that contains at least one optimal assortment. To this end, denote

$$S(z) = (S_1(z), \dots, S_m(z)) \quad (8)$$

where  $S_i(z) = \arg \max_{S_i \in \mathcal{T}_i} V_i(S_i)^{\gamma_i} (R_i(S_i) - z)$ . If multiple maximizers, arbitrarily pick  $S_i(z - \varepsilon)$  or  $S_i(z + \varepsilon)$ .

**Proposition 4** (Candidates for the whole problem). *There exists a set of assortments  $\mathcal{T} = \{S(z), -\infty < z < \infty\}$  which contains at least one optimal assortment, and  $|\mathcal{T}| = O(mn^2)$ .*

*Proof.* Firstly,  $S(Z^*) = (S_1(Z^*), \dots, S_m(Z^*)) \in \mathcal{T}$ . For each nest  $i$ , from the fact

$$S_i(Z^*) \in \arg \max_{S_i \in \mathcal{T}_i} \{V_i(S_i)^{\gamma_i} (R_i(S_i) - Z^*)\},$$

and equivalent relation (6), it follows that  $S_i(Z^*) \in \arg \max_{S_i: |S_i| \leq C_i} \{V_i(S_i)^{\gamma_i} (R_i(S_i) - Z^*)\}$ . Then by Proposition 1,  $S(Z^*)$  is an optimal assortment which belongs to  $\mathcal{T}$ .

Next, we analyze the size of  $\mathcal{T}$ . For each nest  $i$ ,  $S_i(\cdot)$  is the maximizer of  $g_i(\cdot)$  which is piecewise with at most  $|\mathcal{T}_i| - 1$  breakpoints. The union of  $S_i(\cdot)$ 's breakpoints will not exceed  $\sum_{i=1}^m (|\mathcal{T}_i| - 1) = O(mn^2)$ . It is obvious that between any adjacent breakpoints,  $S(\cdot)$  is consistent. Therefore,  $\mathcal{T}$  will contain  $O(mn^2)$  elements.  $\square$

We can see that this result is consistent with Lemma 1: each linear piece of  $G(\cdot)$  corresponds to an assortment and  $G(z)$  is associated with  $S(z)$  as the maximizer for all  $z \in \mathbf{R}$ . In next section, we will design an algorithm to find an optimal assortment by enumerating all elements in  $\mathcal{T}$  efficiently.

#### 4. The algorithm and analysis

Our algorithm is comprised of two stages. The first is to generate  $\{\mathcal{T}_i\}_{i=1}^m$ : the candidate sets for each nest as described in Section 3.1. Then we enumerate all the  $O(mn^2)$  elements in  $\mathcal{T}$ , which is given in Proposition 4, to find an optimal assortment.

##### 4.1. Generating the candidate assortments for an individual nest

Here we propose a simple algorithm to generate  $\mathcal{T}_i$  for nest  $i$ , which is essentially determine the topmost  $C_i$  lines in  $\{f_j(u)\}_{j=1}^n$  for all  $u \in \mathbf{R}$ , where  $f_j(u) = v_{ij}(r_{ij} - u)$ .

Without loss of generality, we assume  $v_{i1} \leq v_{i2} \leq \dots \leq v_{in}$ ; if some  $v_{ij} = v_{i(j+1)}$ , then order  $v_{ij}r_{ij} \leq v_{i(j+1)}r_{i(j+1)}$ . We scan from  $u \rightarrow -\infty$  to  $\infty$ . Initially, for  $u \rightarrow -\infty$ , the order from highest to lowest must be  $(n, n-1, \dots, 1, 0)$ .

As described in Section 3.1, the relative order of lines will be consistent between two consecutive cross-points. If there is a crosspoint  $I_k$  only for two lines  $j_1 > j_2$ , i.e.,  $f_{j_1}(I_k) = f_{j_2}(I_k)$ , their order must be consecutive and should be swapped before and after  $I_k$ , i.e.,  $f_{j_1}(I_k - \varepsilon) > f_{j_2}(I_k - \varepsilon)$  and  $f_{j_1}(I_k + \varepsilon) < f_{j_2}(I_k + \varepsilon)$ .

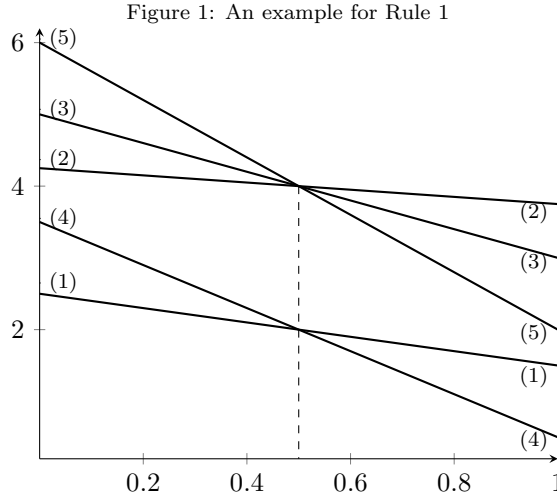


When  $r \geq 2$  lines  $\{j_1, j_2, \dots, j_r\}$  share the same crosspoint  $I_k$ , i.e.,  $f_{j_1}(I_k) = f_{j_2}(I_k) = \dots = f_{j_r}(I_k)$ , their rank must be consecutive and should be reversed before and after  $I_k$  as  $u$  increases. Without loss of generality, assume  $j_1 > \dots > j_r$ , which implies  $f_{j_1}(I_k - \varepsilon) > \dots > f_{j_r}(I_k - \varepsilon)$ . The common crosspoint can be regarded as  $\frac{1}{2}r(r-1)$  overlapped crosspoints. After a series of  $\frac{1}{2}r(r-1)$  swap operations:

$$(j_1, j_2), (j_1, j_3), \dots, (j_1, j_r), (j_2, j_3), \dots, (j_2, j_r), \dots, (j_{r-1}, j_r),$$

the order of  $(j_1, \dots, j_r)$  is reversed, which exactly matches  $f_{j_r}(I_k + \varepsilon) > \dots > f_{j_1}(I_k + \varepsilon)$ . Rule 1 is a summarization.

As an illustration, Figure 1 consists of 5 lines  $\{1, 2, 3, 4, 5\}$  ranked from lowest absolute slope to highest. There are 4 crosspoints at  $u = 0.5$  (3 ones are overlapped). For  $u = 0.5 - \varepsilon$ , the order is  $(5, 3, 2, 4, 1)$ ; after a series of swaption:  $(3, 5), (2, 5), (2, 3), (1, 4)$ , the order becomes  $(2, 3, 5, 1, 4)$ , which is exactly the order at  $u = 0.5 + \varepsilon$ .



**Rule 1** (Tie-breaking). *For any pair of crosspoints  $(u_k, j'_k, j''_k)$  and  $(u_{k+1}, j'_{k+1}, j''_{k+1})$ ,  $u_k \leq u_{k+1}$ ; if  $u_k = u_{k+1}$ , then order  $j'_k \geq j'_{k+1}$ ; if  $u_k = u_{k+1}$  and  $j'_k = j'_{k+1}$ , then order  $j''_k \geq j''_{k+1}$ .*

That is to say, since there are at most  $O(n^2)$  crosspoints, it is sufficient to perform at most  $O(n^2)$  swaps as  $u$  increases. Once the  $C_i$ -th and  $(C_i + 1)$ -th elements swap, we update  $\hat{S}_i(u)$  with the corresponding  $V_i(\hat{S}_i(u))$  and  $R_i(\hat{S}_i(u))$  by removing one and inserting another. Each operation can be done in  $O(1)$  time.

Note that it is not necessary to record all the  $\hat{S}_i(u)$  because  $V_i(\hat{S}_i(u))$  and  $R_i(\hat{S}_i(u))$  are sufficient. Generating and sorting  $O(n^2)$  crosspoints costs  $O(n^2 \log n)$ , and generating all  $V_i(\hat{S}_i(u))$  and  $R_i(\hat{S}_i(u))$  by swapping at most  $O(n^2)$  pairs of elements costs  $O(n^2)$ . Theorem 1 is a summarization of the above analysis.

**Theorem 1.** *If we record  $V_i(\hat{S}_i(u))$  and  $R_i(\hat{S}_i(u))$  instead of each assortment  $\hat{S}_i(u)$ , for each nest  $i$ , Algorithm 1 terminates in  $O(n^2 \log n)$  time with candidate set  $\mathcal{T}_i$ .*

---

**Algorithm 1** Generating candidate set  $\mathcal{T}_i$  for nest  $i$ 


---

**Input:**  $v_{ij}, r_{ij}, \forall j; C_i$

**Output:**  $\mathcal{T}_i$

- 1: Reorder the products such that  $v_{i1} \leq v_{i2} \leq \dots \leq v_{in}$ ; if  $v_{ij} = v_{i(j+1)}$ , then order  $v_{ij}r_{ij} \leq v_{i(j+1)}r_{i(j+1)}$ .
  - 2: Define an order  $R \leftarrow (n, n-1, \dots, 0)$ .
  - 3: Denote  $\hat{S}_i$  as the first  $C_i$  elements of  $R$  with  $V_i(\hat{S}_i)$  and  $R_i(\hat{S}_i)$ ,  $\mathcal{T}_i \leftarrow \{\hat{S}_i\}$ .
  - 4: Generate crosspoints  $\mathcal{I} = \{(u_k, j'_k, j''_k) : f_{j'}(u) = f_{j''}(u), j'_k < j''_k\}$  and sort them by Rule 1.
  - 5: **for**  $k = 1, \dots, p$  **do**
  - 6:     Swap the order of  $(j'_k, j''_k)$  in  $R$ .
  - 7:     **if** the elements at  $C_i$ -th and  $C_{i+1}$ -th order change after  $u_k$  **then**
  - 8:         Denote  $\hat{S}_i$  as the first  $C_i$  elements of  $R$  (if  $\hat{S}_i$  contains 0, ignore 0 and drop all the elements below).
  - 9:         Update  $V_i(\hat{S}_i)$  and  $R_i(\hat{S}_i)$  by replacing the corresponding elements.
  - 10:         $\mathcal{T}_i \leftarrow \mathcal{T}_i \cup \{\hat{S}_i\}$ .
  - 11:     **end if**
  - 12: **end for**
  - 13: **return**  $\mathcal{T}_i$ .
- 

#### 4.2. Enumerate the candidate assortments for the entire system

Here we propose an strongly polynomial algorithm described in Algorithm 2 to solve the problem. Briefly, it is essentially enumerating the candidate set  $\mathcal{T}$  in Proposition 4.

Each loop in Step 1 calculates the expression of  $g_i(\cdot)$  which is stored as  $g_i(z) = \tilde{a}_j^i - \tilde{b}_j^i z, \tilde{u}_j^i \leq z \leq \tilde{u}_{j+1}^i, j = 1, \dots, q$  where  $\tilde{u}_{q+1}^i = \infty$ .

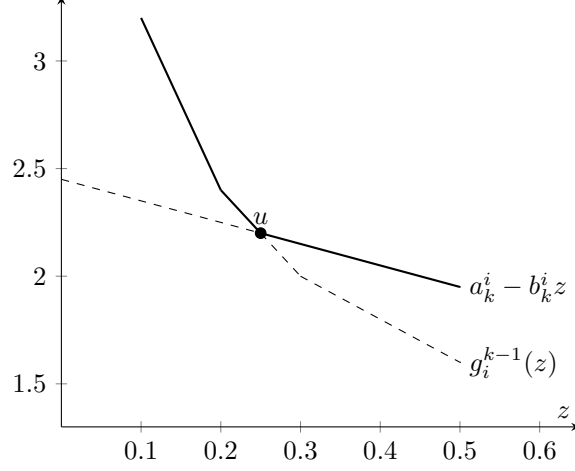
By definition,  $g_i(z)$  is a maximization of a sequence of linear functions simplified as  $g_i(z) := \max_{j=1}^{|\mathcal{L}|} \{a_j^i - b_j^i z\}$ . Without loss of generality, we assume  $b_1^i > \dots > b_{|\mathcal{L}|}^i$  (ranking the lines from highest steepness to lowest) and no  $(a_{j'}^i, b_{j'}^i)$  such that  $\exists b_{j''}^i = b_{j'}^i$  and  $a_{j'}^i < a_{j''}^i$  (line  $j'$  is redundant). With a slight abuse of notation, we define  $g_i^k(z) = \max_{j=1}^k \{a_j^i - b_j^i z\}$ . For  $k \geq 2$ , the recursive formulation is  $g_i^k(z) = \max\{g_i^{k-1}(z), a_k^i - b_k^i z\}$ . The iteration ends up with  $g_i(z) = g_i^{|\mathcal{L}|}(z)$ .

For  $k \geq 2$ , we know that the (sub)gradient of  $g_i^{k-1}(z)$  for any  $z \in \mathbf{R}$  is strictly less than  $-b_k^i$ . There must be a breakpoint  $u$  such that  $g_i^{k-1}(u) = a_k^i - b_k^i u, z < u \Leftrightarrow g_i^{k-1}(z) > a_k^i - b_k^i z$  and  $z > u \Leftrightarrow g_i^{k-1}(z) < a_k^i - b_k^i z$  (see Figure 2 for an illustration). Therefore  $g_i^k(z)$  can be formulated by some leftmost linear piece of  $g_i^{k-1}(\cdot)$  plus a new linear piece

$$g_i^k(z) = \begin{cases} g_i^{k-1}(z), & z \leq u \\ a_k^i - b_k^i z, & z \geq u \end{cases}$$

After repeatedly delete the rightmost linear piece until we find the intersection point  $u$ , we create a new linear piece for  $z \geq u$  and the function is then updated to  $g_i^k(\cdot)$ .

Figure 2: The maximization of  $g_i^{k-1}(z)$  and  $a_k^i - b_k^i z$



In Step 2, we enumerate all assortments in  $\mathcal{T}$  to find the maximal expected revenue. Denote  $A(z) = \sum_{i=1}^m V_i(S_i(z))^{\gamma_i} R_i(S_i(z))$  and  $B(z) = v_0 + \sum_{i=1}^m V_i(S_i(z))^{\gamma_i}$ . By definition, the expected revenue for assortment  $S_i(z)$  is  $A(z)/B(z)$ . Also,  $G(z) = v_0 z + \sum_{i=1}^m g_i(z) := A(z) - B(z)z$ . To calculate the maximal expected revenue, the algorithm only needs attributes  $A(z)$  and  $B(z)$  for each  $S(z)$ .

The set of  $G(\cdot)$ 's breakpoints consists of all breakpoints of  $g_i(\cdot)$  for all  $i = 1, \dots, m$ ; also  $A(\cdot)$  and  $B(\cdot)$ . The iterative expression for  $A(\cdot)$  is

$$A(-\infty) = A(-\infty) + \sum_{i=1}^m \tilde{a}_1^i,$$

$$A(z_+) = A(z_-) + \sum_{(i,j): \tilde{u}_j^i = z} (\tilde{a}_j^i - \tilde{a}_{j-1}^i),$$

and  $B(\cdot)$  is similar. We gather and sort all breakpoints to  $u_1 \leq \dots \leq u_{|\Delta|}$ . For each  $u_i$ , we update  $(A, B)$  to  $(A(u_i), B(u_i))$  from previous one. The algorithm finds the largest expected revenue  $Z^* = \max_{z \in \mathbf{R}} \{A(z)/B(z)\}$  by essentially enumerating all elements in  $\mathcal{T}$ . By the way, it is simple to find an optimal assortment given  $Z^*$ .

Next we discuss the time complexity except for generating  $\{\mathcal{T}_i\}_{i=1}^m$ .

The first part: For each nest  $i$ , sort  $|\mathcal{L}^i| = O(n^2)$  elements costs  $O(n^2 \log n)$ ; each iteration will create a new piece and each linear piece can be delete at most once. In total, the first part costs  $O(mn^2 \log n)$ .

The second part: the sorting, which is essentially merging  $m$   $O(n^2)$ -sized ordered lists, can be done in  $O(mn^2 \log m)$ ; the enumeration costs  $O(mn^2)$ .

Theorem 2 is a summary of the above analysis.

**Theorem 2.** *Algorithm 2 terminates with an optimal assortment in  $O(mn^2 \log m)$  time.*

The workflow to solve the optimization problem is: (1) Generate  $\{\mathcal{T}_i\}_{i=1}^m$  by running Algorithm 1  $m$

times, which costs  $O(mn^2 \log n)$  time; (2) Run Algorithm 2 to determine the maximal expected revenue and find an optimal assortment. The total running time is  $O(mn^2 \log mn)$ .

## 5. Computational Experiments

In this section, we conduct some computational experiments to show the efficiency of our algorithm. In particular, we compare Algorithm 2 with directly solving linear program formulation (7). In our tests, it is solved by IBM ILOG CPLEX 12.6 (64-bit version) via the ILOG Concert API. The base frequency of the CPU for the tests is 2.10GHz.

We randomly generate 10 problem instances for each setting. The number of nest  $m$  and the products in each nest  $n$  is specified in each setting. For all product  $ij$ , the weight  $v_{ij}$  is i.i.d. random variable uniformly from  $[0.1, 10.0]$  and the revenue  $r_{ij}$  is i.i.d. uniformly from  $[0, 10]$ . While the weight of no-purchase  $v_0 = 1$ . The dissimilarity parameters  $\gamma_i = 0.5$  for all  $i$  and the constraints  $C_i = n/2$  for  $i$ .

The results are shown in Table 1. Note that the experiment is memory-consuming. Our Algorithm 2 is optimized to reduce the memory consumption and therefore performs well for all the instances. In particular, Table 1 suggests that Algorithm 2 is able to solve a instance with 200000 nests and each nest includes 200 products within a minute. By contrast, CPLEX consumes more time and more memory when the problem size increases. In summary, the results suggest that our algorithm is an efficient approach when the problem size is large.

Table 1: Comparison of the running time in seconds for Algorithm 2 and CPLEX

$m \backslash n$	10		20		50		100		200	
	Alg. 2	CPLEX	Alg. 2	CPLEX	Alg. 2	CPLEX	Alg. 2	CPLEX	Alg. 2	CPLEX
100	0.00	0.01	0.00	0.02	0.01	0.06	0.00	0.06	0.01	0.11
200	0.00	0.03	0.00	0.06	0.01	0.07	0.01	0.12	0.02	0.25
500	0.00	0.17	0.00	0.13	0.02	0.25	0.02	0.46	0.05	1.04
1000	0.00	0.23	0.01	0.36	0.03	0.57	0.05	1.01	0.10	2.46
2000	0.01	0.40	0.02	0.55	0.05	1.05	0.11	2.02	0.23	4.22
5000	0.03	0.78	0.05	1.24	0.14	2.62	0.30	5.16	0.62	10.90
10000	0.05	1.94	0.11	2.84	0.30	5.72	0.64	13.00	1.35	23.59
20000	0.12	5.65	0.24	7.68	0.64	18.57	1.37	29.03	2.90	52.02
50000	0.32	27.34	0.64	46.55	1.75	61.71	3.75	88.38	8.13	150.42
100000	0.67	122.72	1.38	136.70	3.89	166.49	8.40	217.51	18.26	344.78
200000	1.46	420.28	3.07	442.55	8.59	516.79	18.86	643.24	40.56	1040.59

## Acknowledgement

We are grateful to Jiawei Zhang for introducing us this problem. We thank Bo Jiang and Yi Xu for the discussion and useful suggestions.

## References

- [1] M. E. Ben-Akiva, S. R. Lerman, Discrete choice analysis: theory and application to travel demand, Vol. 9, MIT press, 1985.
- [2] J. M. Davis, G. Gallego, H. Topaloglu, Assortment optimization under variants of the nested logit model, *Oper. Res.* 62(2) (2014), 250-273.
- [3] A. G. Kök, M. L. Fisher, R. Vaidyanathan, Assortment planning: Review of literature and industry practice. in: N. Agrawal, S. A. Smith (eds.), *Retail Supply Chain Management*, Springer US, 2015, pp. 175-236.
- [4] J. B. Feldman, H. Topaloglu, Capacity Constraints Across Nests in Assortment Optimization Under the Nested Logit Model, *Oper. Res.* 63(4) (2015), 812-822.
- [5] G. Gallego, H. Topaloglu, Constrained assortment optimization for the nested logit model, *Manage. Sci.* 60(10) (2014), 2583-2601.
- [6] G. Li, P. Rusmevichientong, A greedy algorithm for the two-level nested logit model, *Oper. Res. Lett.* 42(5) (2014), 319-324.
- [7] D. McFadden, Conditional logit analysis of qualitative choice behavior, in: P. Zarembka (ed.), *Frontiers in Econometrics*, Academic Press, 1973, pp. 105-142.
- [8] D. McFadden, Modelling the choice of residential location, in: A. Karlqvist, L. Lundqvist, F. Snickars, J. Weibull (eds.), *Spatial Interaction Theory and Planning Models*, North-Holland, Amsterdam, 1978, pp. 75-96.
- [9] P. Rusmevichientong, Z.-J. M. Shen, D. B. Shmoys, Dynamic assortment optimization with a multinomial logit choice model and capacity constraint, *Oper. Res.* 58(6) (2010), 1666-1680.
- [10] K. Talluri, G. van Ryzin, Revenue management under a general discrete choice model of consumer behavior, *Manage. Sci.* 50(1) (2004), 15-33.
- [11] H. C. Williams, On the formation of travel demand models and economic evaluation measures of user benefit, *Environ. Plan. A*, 9(3) (1977), 285-344.

---

**Algorithm 2** Combinatorial algorithm for disjoint-cardinality constraints

---

**Input:**  $\{\mathcal{T}_i\}_{i=1}^m$  generated by Algorithm 1.

**Output:** Optimal expected revenue  $Z^*$  and optimal assortment  $(S_1^*, S_2^*, \dots, S_m^*)$ .

```
1: Step 1: Iteratively calculating the expression of  $g_i(\cdot)$ 
2: for  $i = 1, \dots, m$  do
3:   Let  $\mathcal{L}^i = \{(a_j^i, b_j^i) := (V_i(S_i)^{\gamma_i} R_i(S_i), V_i(S_i)^{\gamma_i}), \forall S_i \in \mathcal{T}_i\}$ .
4:   Sort  $\mathcal{L}^i$  to ensure  $b_j^i > b_{j+1}^i, \forall j$  while drop all  $(a_{j'}^i, b_{j'}^i)$  such that  $\exists b_{j''}^i = b_{j'}^i$  and  $a_{j'}^i < a_{j''}^i$ .
5:    $q^i \leftarrow 1, (\tilde{u}_1^i, \tilde{a}_1^i, \tilde{b}_1^i) \leftarrow (-\infty, a_1^i, b_1^i)$ .
6:   for  $j = 2, \dots, |\mathcal{L}^i|$  do
7:     while  $a_j^i - b_j^i \tilde{u}_{q^i}^i \geq \tilde{a}_{q^i}^i - \tilde{b}_{q^i}^i \tilde{u}_{q^i}^i$  do
8:        $q^i \leftarrow q^i - 1$ .
9:     end while
10:     $q^i \leftarrow q^i + 1, (\tilde{u}_{q^i}^i, \tilde{a}_{q^i}^i, \tilde{b}_{q^i}^i) \leftarrow \left( \frac{\tilde{a}_{q^i}^i - a_j^i}{\tilde{b}_{q^i}^i - b_j^i}, a_j^i, b_j^i \right)$ .
11:  end for
12: end for
13: Step 2: Enumerating  $\mathcal{T}$  to find the maximal expected revenue
14:  $A \leftarrow \sum_{i=1}^m \tilde{a}_1^i, B \leftarrow v_0 + \sum_{i=1}^m \tilde{b}_1^i$ .
15:  $\Delta \leftarrow \bigcup_{i=1}^m \left\{ (u, \Delta a, \Delta b) := (\tilde{u}_j^i, \tilde{a}_j^i - \tilde{a}_{j-1}^i, \tilde{b}_j^i - \tilde{b}_{j-1}^i), \forall j = 2, \dots, q^i \right\}$ , sort  $\Delta$  to ensure  $u_i \leq u_{i+1}, \forall i$ .
16:  $Z^* \leftarrow A/B$ .
17: for  $i = 1, \dots, |\Delta|$  do
18:    $A \leftarrow A + (\Delta a)_i, B \leftarrow B + (\Delta b)_i$ .
19:    $Z^* \leftarrow \max\{Z^*, A/B\}$ .
20: end for
21:  $S_i^* = \arg \max_{S_i \in \mathcal{T}_i} V_i(S_i)^{\gamma_i} (R_i(S_i) - Z^*)$ .
22: return Optimal expected revenue  $Z^*$  and optimal assortment  $(S_1^*, S_2^*, \dots, S_m^*)$ .
```

---